Fault Diagnosis of Embedded Software using Program Spectra*

Peter Zoeteweij[1] Rui Abreu[1] Rob Golsteijn[2]
Arjan J.C. van Gemund[1]

[1]Embedded Software Lab,
Software Engineering Research Group, TU Delft
{p.zoeteweij,r.f.abreu,a.j.c.vangemund}@tudelft.nl

[2]NXP
rob.golsteijn@nxp.com

In this paper we present a technique to automatically diagnose errors detected during software testing. With diagnosis we mean localization of the fault that causes these errors. Strictly speaking, this belongs to the debugging phase, rather than to the testing phase of the software development cycle, but the technique is well suited for integration with (automated) testing, and enhances the information that can be extracted from this phase.

We assume that testing involves a number of different runs (or transactions, usage scenario's, etc.) of the same software. Per run, we record a so-called *program spectrum* (see, e.g., [2]). The particular form of program spectrum that we are interested in is an array of Boolean flags that tells us which parts of the software were active during a run. This can be measured at different levels of granularity, for example, at component, class, or function level, but also at the level of individual statements, or blocks of code. Together, the program spectra for a series of test runs form a binary matrix, whose row vectors correspond to the runs, and whose column vectors tell us in which runs a particular part of the software was active (see Figure 1).

During testing, errors are detected in some of the runs. This information constitutes another column vector, the error vector, which can be seen as to correspond to a hypothetical part of the software that is responsible for the detected errors. The assumption is that the actual parts of the software whose column vectors resemble the error vector most, have a high probability of containing the fault. Basic data clustering techniques can be applied to

*This work has been carried out as part of the TRADER project under the responsibility of the Embedded Systems Institute, and is partially supported by the Netherlands Ministry of Economic Affairs under the BSIK03021 program.

$$
\begin{array}{cc}
 & \begin{array}{cc} N \text{ parts} & \text{errors} \end{array} \\
M \text{ runs} & \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{M1} & x_{M2} & \cdots & x_{MN} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_M \end{bmatrix}
\end{array}
$$

Figure 1: The ingredients of fault diagnosis

quantify this resemblance, which gives us a ranking of likely fault locations. Recent studies on a benchmark set of faults (see, e.g., [1, 3]) indicate that the use of this ranking can significantly reduce the manual debugging effort, in terms of the amount of code that must be investigated.

Because of their compact representation (compared to, e.g., execution traces), program spectra are an attractive technique in the resource-constrained environment that is typical for the development of embedded software. As a proof of concept, we applied the technique to the control software of a particular product line of television sets. In these experiments we diagnosed two realistic errors, one existing, and one that was seeded to reproduce a problem with another product line. For both errors, we obtained a highly accurate diagnosis: in one case the actual fault came second in a ranking of 315 logical threads, and in the other case, we were able to identify a single block of code, containing one statement, out of 60K blocks, in approximately 450K lines of C code.

The essential characteristics of our implementation are: (1) automated source code instrumentation for recording the spectra, (2) limited caching of spectra for past transactions, and (3) processing of cached spectra on a low-priority thread. In our case, we processed the spectra off-line, transporting them via a serial connection to a PC. However, we believe that to a large extent, the analysis described above can be done incrementally, on the TV, which would allow us to reduce the need for caching spectra.

To get a better insight in the possibilities and limitations of using program spectra to diagnose software faults, we are currently investigating the influence of various external factors on the quality of the diagnosis. One example of such a factor is the percentage of all occurring errors that is detected during testing, which can be influenced by using different error detection mechanisms, such as, e.g., array bounds checking and assertions. For future research we plan to investigate if the approach needs to be extended for the case of multiple faults, and how additional knowledge about the software, such as data and control-flow dependencies, can best be put to use. Our work will culminate into a C compiler that instruments the code under test such that error detection, spectral profiling, and diagnosis are all embedded in the application.

References

[1] R. Abreu, P. Zoeteweij, A.J.C. van Gemund. An evaluation of similarity coefficients for software fault localization. In *Proceedings of PRDC'06*, IEEE Press, 2006. To appear.

[2] M.J. Harrold, G. Rothermel, R. Wu, and L. Yi. An empirical investigation of program spectra. In *ACM SIGPLAN Notices*, 33(7):83–90, 1998.

[3] J. A. Jones and M. J. Harrold. Empirical evaluation of the Tarantula automatic fault-localization technique. In *Proceedings of ASE'05*, pages 273–282, ACM, 2005.

About the Authors

Peter Zoeteweij works in the Software Engineering Research Group at Delft University of Technology. He holds an MSc. from Delft University of Technology, and a PhD. from the University of Amsterdam, both in computer science. Before his PhD., Peter worked for several years as a software engineer for Logica (now LogicaCMG), mainly on software for the oil industry.

Rui Abreu is a PhD. student at the Embedded Software Lab within the Software Engineering Research Group at Delft University of Technology. He holds an MSc. in Computer Science and Systems Engineering from Minho University, Portugal. Through his thesis work at Siemens R&D Porto, and professional internship at Philips Research, he acquired industrial experience in the area of embedded systems.

Rob Golsteijn holds an MSc. in Computing Science from Eindhoven University of Technology and completed the two years' post-graduate Software Technology program from the Stan Ackermans Institute. Rob now works for NXP, formerly known as Philips Semiconductors, and has experience in embedded software development of television platforms and products. Rob is currently working as a member of an industrial research project focusing on reliability of resource-constrained consumer devices.

Arjan J.C. van Gemund holds a BSc. in physics, and an MSc. (cum laude) and PhD. (cum laude) in computer science, all from Delft University of Technology. He has held positions at DSM and TNO, and currently serves as a full professor at the Electrical Engineering, Mathematics, and Computer Science Faculty of Delft University of Technology, heading the Embedded Software Lab within the Software Engineering Research Group.